Anisotropic Point-Based Fusion

Damien Lefloch
Computer Graphics Group
University of Siegen
Email: damien.lefloch@uni-siegen.de

Tim Weyrich
Dept of Computer Science
University College London
Email: t.weyrich@ucl.ac.uk

Andreas Kolb
Computer Graphics Group
University of Siegen
Email: andreas.kolb@uni-siegen.de

***Abstract*—We propose a new real-time framework which efficiently reconstructs large-scale scenery by accumulating anisotropic point representations in combination with memory efficient representation of point attributes. The reduced memory footprint allows to store additional point properties that represent the accumulated anisotropic noise of the input range data in the reconstructed scene. We propose an efficient processing scheme for the extended and compressed point attributes that does not obstruct real-time reconstruction. Furthermore, we evaluate the positive impact of the anisotropy handling on the data accumulation and the 3D reconstruction quality.**

I. Introduction and Prior Work

Following the seminal paper by Rusinkiewicz et al. [1], and further popularized through the availability of affordable structured-light and Time-of-Flight depth cameras and the prominent KinectFusion system [2], interactive real-time scene acquisition from hand-held depth cameras has developed much momentum, enabling applications in ad-hoc object acquisition, augmented reality and other fields.

Previous work has mainly focused on representing the accumulated model using implicit volumetric voxel grids in which truncated distance fields are accumulated in order to store the probability of a voxel being at the observed surface. As this representation is rather inefficient in terms of spatial adaptivity and scalability, various approaches have been proposed to overcome this restriction, e.g., by adopting the spatial position and orientation of the voxel grid [3] or by hashing and book-keeping of smaller voxel bricks [4]. An alternative point-based representation has been presented by Keller et al. [5].

Common to all these systems is a three-stage process, consisting of the following components; see also Fig. 1:

1) **Depth Map Preprocessing:** The range map delivered by the Kinect camera is preprocessed, e.g., using bilateral filtering, and additional data such as normals are estimated for each range map pixel.
2) **Camera Pose Estimation:** Based on the current observation and the so far accumulated model, the camera pose is estimated using an Iterative Closest Point (ICP) approach [6].
3) **Depth Map Fusion:** In this step the registered input range map is accumulated into the existing model representation.

One aspect that has insufficiently been addressed so far is the *anisotropic nature* of the input data. The spatial uncertainty of an individual pixel of the input range map is determined by two factors:

a) the *lateral pixel extent* which is given by the lateral resolution of the camera chip and the intrinsic parameters of the camera, i.e, focal length, principal point and lens distortion, in combination with the depth value, i.e. the distance from the camera, and
b) the *depth noise* of the sensor, which itself strongly depends on the underlying range measurement principle.

There are already some works on noise models for range devices, e.g., for Time-of-Flight (ToF) cameras such as the new generation Kinect One. Falie and Buzuloiu [7] present a noise model based one phenomenological considerations, which predicts a range error as a function of the amplitude and the distance value of a specific pixel. For an overview of denoising approaches for ToF cameras, refers to the survey of Lenzen et al. [8]. Often simple Gaussian noise models are assumed, e.g., in the context of motion capturing [9].

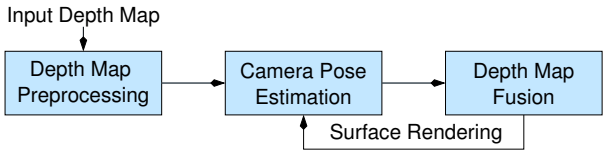

Fig. 1. The KinectFusion pipeline.

Maier-Hein et al. [10] introduce a method in order to improve ICP-based registration of ToF-based range maps with respect to a given polygonal model in the context of medical applications. However, as of yet, the anisotropy of the input data has not been considered in the context of real-time scene acquisition.

In this paper we present a new real-time framework for efficient reconstruction of large-scale scenery incorporating the anisotropy of the input data. Our system uses an enhanced point-based representation similar to Keller et al. [5] which is capable of handling anisotropy in the depth map fusion step; see Fig. 1. Our main contributions are:

- a novel *symmetric anisotropic distance measure* that is applied to establish more robust correspondences between input and model points in the fusion step, and
- a novel *anisotropy-aware fusion technique* for accumulation of anisotropic input data into the model,
- a *data compression scheme* for point-based model representation implying an efficient storage of attributes per-point.

Furthermore, we present a solid evaluation on both, the

data compression scheme and the anisotropic accumulation approach, and their impact on the reconstruction quality.

II. Overview Point-based Fusion

Our approach shares many design aspects with the point-based KinectFusion method proposed by Keller et al. [5]. The main components of the online, point-based scene reconstruction are the following stages; see Fig. 1.

a) Depth Map Preprocessing: Denoting a pixel of the input frame as $\boldsymbol{u} = (x,y)^\top \in \mathbb{R}^2$, an initial 2D *vertex map* $\mathcal{V}_i(\boldsymbol{u}) \in \mathbb{R}^3$ is generated from the depth map $\mathcal{D}_i(\boldsymbol{u}) \in \mathbb{R}$ for frame i by applying the inverse of the intrinsic camera matrix $\boldsymbol{K}_i$. Additional information is computed and stored in *attribute maps*: a normal map $\mathcal{N}_i$, the point radius map $\mathcal{R}_i$.

b) Depth Map Fusion: Given a valid camera pose and the corresponding vertex and attribute maps, the geometric data is fused into the *global model*. The global model consists of a simple list of attributed 3D points. Model points evolve from *unstable* to *stable* status based on the confidence they gathered. The confidence essentially counts how often a point has been observed by the sensor. Data fusion first *projectively associates* points in the input depth map with the points in the global model by rendering the model from the current camera position as an *index map*. If point partners are found, the input point is merged with the best matching model point using a weighted average for the point position and attributes. If no merge partner is found for an input point, the new point is added to the global model as an unstable point. The global model is continuously cleaned up over time to remove outliers due to visibility and temporal constraints, removing isolated observations that have not been confirmed by further observations over a specific period of time.

c) Camera Pose Estimation: All stable model points are passed to the visualization stage which reconstructs a dense representation of the model's surface including the associated attributes, i.e., normal and size, using a surface splatting technique. To estimate the 6DoF camera pose, the model points are projected from the previous camera pose, and a pyramidal dense iterative-closest-point (ICP) [2] alignment is performed using the synthesized *model map* and the input depth map. The resulting relative transformation rigidly links the previous to the new global camera pose.

Regarding the rendering, which is not further discussed in this paper, we stick to the simple splat surfel rendering used in Keller et al. [5] which trades off the rendering quality in order to achieve a fast synthetic view generation. For visual user feedback, we use a simple Phong illumination model coupled with a fast approximation of ambient occlusion known as Screen-Space Ambient Occlusion (SSAO) [11].

III. Anisotropic Point-based Fusion

We introduce a new reconstruction framework that stores as an additional per-point property the 3×3, anisotropic noise covariance matrix $\Sigma(\boldsymbol{u})$.

A. Anisotropy

So far, real-time reconstruction methods with range maps have ignored the anisotropic nature of the range data. The anisotropy results from the fact, that the reliability of a 3D point in a range map is much higher in lateral direction than in axial direction, as the lateral uncertainty is only limited by the pixel size and, due to the perspective mapping, by the distance. The axial uncertainty is defined by the noise of the acquisition device, i.e., the Kinect camera in our case, which, for example, increases for larger object-to-camera distances. Maier-Hein et al. [10] model the standard deviation as a function over distance. We use the model for the Kinect camera proposed by Nguyen et al. [12] in order to compute the variance of the noise based on the z-distance.

Given a covariance matrix $\Sigma_{\boldsymbol{p}}$ for a point $\boldsymbol{p} \in \mathbb{R}^3$, the *Mahalanobis distance* of any other point $\boldsymbol{q} \in \mathbb{R}^3$ can be calculated based on the inverse of the covariance matrix $\Sigma_{\boldsymbol{p}}^{-1}$, which is also called *reliability matrix*:

$$d_{\boldsymbol{p},\Sigma}(\boldsymbol{q}) = \sqrt{(\boldsymbol{q}-\boldsymbol{p})^\top \Sigma_{\boldsymbol{p}}^{-1} (\boldsymbol{q}-\boldsymbol{p})}\,.$$

We directly store the symmetric 3×3 reliability matrix $\Sigma_{\boldsymbol{p}}^{-1}$ leading to 6 additional values per point.

Similar to Maier-Hein et al. [10], we build the data association before data fusion using our anisotropic model (see Sec. II for a short description of the data association). While they use the Malahanobis distance based on the inverse of the sum of covariance matrix $(\Sigma_{\boldsymbol{p}} + \Sigma_{\boldsymbol{q}})^{-1}$, we minimize the sum of both Malahanobis distances $d_{\boldsymbol{p},\Sigma}(\boldsymbol{p}-\boldsymbol{q}) + d_{\boldsymbol{q},\Sigma}(\boldsymbol{p}-\boldsymbol{q})$ in order to choose the best associated corresponding pair for accumulation. The main reason of this approach is performance. As we store the reliability matrix $\Sigma_{\boldsymbol{p}}^{-1}$, applying Maier-Hein et al. [10] would require three additional matrix inversions per point-pair comparison. We conducted several experiments to compare our simple minimization to the one proposed in [10]. All experiments were leading to the same result, i.e, same points pairs. This validates our choice to keep our data association for a better efficiency.

B. Anisotropic Fusion

The accumulation of range data in the anisotropic case has to consider the non-uniformity of distance measurements given by the depth sensor. Similar to the geometric fusion, the anisotropic noise model should be refined over time. Therefore, the geometric and anisotropic fusion procedures have to be reformulated by convex combinations for accumulating of the point's mean and the accumulation of the reliability matrix.

Considering two different points $\boldsymbol{p}_i$ with covariance matrices $\Sigma_{\boldsymbol{p}_i}$, $i = 1,2$, and point $\boldsymbol{q}$ lying on the line segment between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, a meaningful definition of an *anisotropic split ratio* β of $\boldsymbol{q}$ with respect to $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is given by

$$\begin{aligned}\boldsymbol{q} &= \frac{d_{\boldsymbol{p}_2,\Sigma_2}(\boldsymbol{q})}{d_{\boldsymbol{p}_1,\Sigma_1}(\boldsymbol{q}) + d_{\boldsymbol{p}_2,\Sigma_2}(\boldsymbol{q})}\,\boldsymbol{p}_1 + \frac{d_{\boldsymbol{p}_1,\Sigma_1}(\boldsymbol{q})}{d_{\boldsymbol{p}_1,\Sigma_1}(\boldsymbol{q}) + d_{\boldsymbol{p}_2,\Sigma_2}(\boldsymbol{q})}\,\boldsymbol{p}_2 \\ &= (1-\beta)\boldsymbol{p}_1 + \beta\boldsymbol{p}_2, \text{ with } \beta = \frac{d_{\boldsymbol{p}_1,\Sigma_1}(\boldsymbol{q})}{d_{\boldsymbol{p}_1,\Sigma_1}(\boldsymbol{q}) + d_{\boldsymbol{p}_2,\Sigma_2}(\boldsymbol{q})}\,. \quad (1)\end{aligned}$$

Within the context of point-based fusion, the points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{q}$ may refer to the model point, the corresponding input point and the resulting merged point, respectively.

Since $\boldsymbol{q}$ is the resulting merged point, we need to reformulate the anisotropic split ratio β as given in Eq. (1). Defining $\boldsymbol{q}$ as affine combination $\boldsymbol{q} = (1-\alpha)\boldsymbol{p}_1 + \alpha\boldsymbol{p}_2$ for some $\alpha \in [0,1]$ and exploiting, that the Mahalanobis distance simply scales the isotropic distance values for a given direction, from Eq. (1) we get

$$\begin{aligned}\beta &= \frac{d_{\boldsymbol{p}_1,\Sigma_1}((1-\alpha)\boldsymbol{p}_1 + \alpha\boldsymbol{p}_2)}{d_{\boldsymbol{p}_1,\Sigma_1}((1-\alpha)\boldsymbol{p}_1 + \alpha\boldsymbol{p}_2) + d_{\boldsymbol{p}_2,\Sigma_2}((1-\alpha)\boldsymbol{p}_1 + \alpha\boldsymbol{p}_2)} \\ &= \frac{\alpha d_{\boldsymbol{p}_1,\Sigma_1}(\boldsymbol{p}_2)}{(1-\alpha)d_{\boldsymbol{p}_2,\Sigma_2}(\boldsymbol{p}_1) + \alpha d_{\boldsymbol{p}_1,\Sigma_1}(\boldsymbol{p}_2)} \,. \end{aligned} \tag{2}$$

Inverting Eq. (2), we get the proper affine weight α that needs to be applied to achieve the desired anisotropic split ratio β

$$\alpha = \frac{\beta d_{\boldsymbol{p}_1,\Sigma_1}(\boldsymbol{p}_2)}{(1-\beta)d_{\boldsymbol{p}_2,\Sigma_2}(\boldsymbol{p}_1) + \beta d_{\boldsymbol{p}_1,\Sigma_1}(\boldsymbol{p}_2)} \,.$$

Analogously, the *anisotropic split ratio* β is used to accumulate the point normals.

Regarding the model accumulation of the covariance represented in the same coordinate system, we apply the approach proposed by Kerl et al. [13]. They perform the covariance accumulation by adding the reliability: given input and model covariance matrices Σ_i^{in} and Σ_i^{mod} for a corresponding input and model point for frame i, respectively, the fused covariance matrix reads as

$$\widehat{(\Sigma_i^{\text{mod}})}^{-1} = (\Sigma_i^{\text{mod}})^{-1} + (\Sigma_i^{\text{in}})^{-1} \,. \tag{3}$$

Note that in order to transform the covariance matrix $\Sigma_{i_{\text{mod}}}^{\text{mod}}$ to the same coordinate system of the input frame Σ_i^{mod}, we have to apply the following transformation:

$$\Sigma_i^{\text{mod}} = (\boldsymbol{R}^{\top}{}_{i\to\text{WC}} \cdot \boldsymbol{R}_{i_{\text{mod}}\to\text{WC}})\Sigma_{i_{\text{mod}}}^{\text{mod}}(\boldsymbol{R}^{\top}{}_{i\to\text{WC}} \cdot \boldsymbol{R}_{i_{\text{mod}}\to\text{WC}})^{\top} \,,$$

with $\boldsymbol{R}_{i_{\text{mod}}\to\text{WC}}$ and $\boldsymbol{R}_{i\to\text{WC}}$ referring to the rotational part of the transformations $\boldsymbol{T}_{i_{\text{mod}}\to\text{WC}}$ and $\boldsymbol{T}_{i\to\text{WC}}$ to pass from local to world coordinates (WC), respectively.

IV. Implementation

notation: In the following, we adopt the data type nomenclature given by [14] where `uint`b refers to a positive integer with b bits representing integers on $\left[0, 2^b - 1\right]$ and `float`b is the floating-point representation with b bits in total describing sign, mantissa and exponent.

In order to store the symmetric reliability matrix $(\Sigma_i^{mod})^{-1}$ for each point inside our model representation, an efficient reduction of memory footprint for the point properties is required to preserve the scalability of the overall acquisition system. Salas-Moreno et al. [15] propose a point-based accumulation model which directly reduces the total number of points by efficiently encoding points belonging to the same planar surface using a new planar representation. The method was shown to be robust and efficient, but it is mainly designed for indoor scenes, which comprise many planar regions.

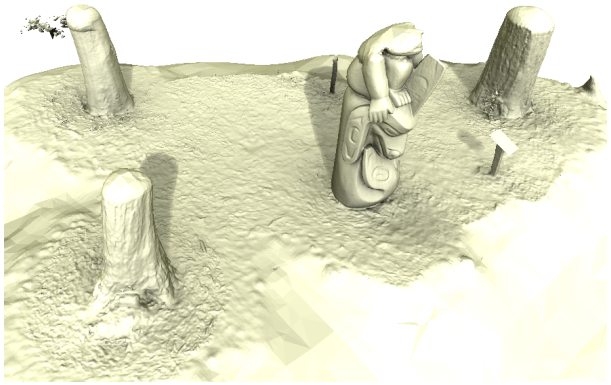

Fig. 2. Advanced rendering of the extracted surface mesh given by our point-based reconstruction framework (*Totempole* scene).

Since we would like to reduce the storage cost of the point-based fusion framework for any type of data set (see, for example, Fig. 2 for a very large scene from Zhou and Koltun [16]), we decide to directly compress the point properties. A naive way to store all required point properties that is, position, normal, radius, confidence counter and timestamp, would require 9 `float32` scalars leading to a total of 288 bits per point.

To compress the surface normal property, we adopt the method proposed by Praun and Hoppe [17] designed to compress unit vectors efficiently. This method first maps the unit sphere to a unit octahedron that is later on unfolded to the $z = 0$ plane. This method is known as one of the best approaches to compress unit vectors rapidly and robustly. Recently a survey of unit vector compression by Cigolle et al. [14] shows that the simple octahedron compression (non-numerically optimized) using 16 bits encoding (i.e. `enc16`) for both texture coordinates leads to a mean error angle of 0.37709° whereas the one using 32 bits (i.e. `enc32`) leads to a mean error of 0.00131°. At a first glance, a mean error of less than half a degree might appear negligible, we show that the impact of the 8 bits encoding on the accumulation significantly coarsens the final reconstructed model. Fig. 3 gives a visual comparison of different encoding schemes applied to the *Totempole* data set.

The point position is also compressed by partially adopting the same method. First, all model points are expressed in their local coordinate referring to the camera position from which they were last observed. The original point based fusion method [5] represents the model points in world coordinate. Practically, once a fusion of an input point and model point occurs, the new average model point will be represented in the camera coordinate system of the current input frame i. This representation enables us to encode the vertices using their viewing direction and their polar distance. We can use the same procedure to encode the viewing ray as we apply to the normal vector. We further assume that consumer depth cameras only provide range measurements up to a maximum radial distance of 10 meters with millimeter precision. Thus, we can store the polar distance ρ expressed in meters in one `uint16` scalar

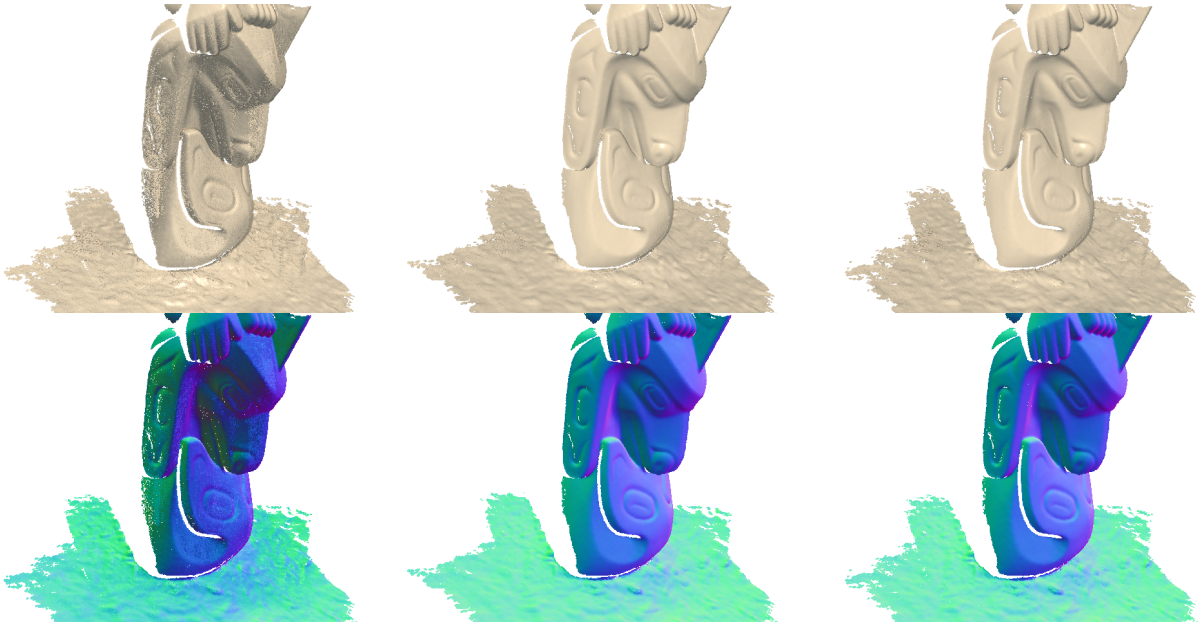

Fig. 3. Comparison of three different compression schemes at frame 380 of the *Totempole* scene using our SSAO surfel splatting. The compression `enc16` (left) leads to a coarser model compared to the original, uncompressed version (center). However, `enc32` (right) shows negligible visual difference. The second row refers to the color coded normal maps.

applying the following encoding $\rho_e = \lfloor 6553.5 \cdot \rho \rceil$[1]. Our vertex position encoding requires only $32+16=48$ bits per model point in contrast to the usual $3 \cdot 32 = 96$ bit storage.

The drawback of this method is that it requires to save all camera pose transformations $\boldsymbol{T}_{i \to \mathrm{WC}}$ in order to transform all model points to common world coordinates. Nevertheless, in Sec. V we show that this additional storage requirement has very little influence on the achieved compression ratio.

The remaining properties, i.e., radius, timestamp and confidence counter, are also encoded using a simple quantization. We store the timestamp t in a `uint16` scalar, leading to a maximum frame id of 65535. Using a 30 Hz camera, it represents more than 30 minutes acquisition time which is sufficient for most applications. The confidence counter is described as a `uint8` scalar since it is usually clamped to a maximum value of 255 to allow for adaptation to changes in the scene [2]. Similar to Weise et al. [18], the radius property is computed by using the following formulation

$$r = \delta_{\mathrm{pix}} \cdot \frac{\max(s_x, s_y)}{f} \cdot \frac{d}{\langle \boldsymbol{n}, [0,0,1]^\top \rangle} \,, \tag{4}$$

where f, s_x and s_y are given by the intrinsic parameters of the camera and represent the focal length and the pixel size in horizontal and vertical directions, respectively. δ_{pix} represents the half of the pixel's diagonal $\sqrt{2}/2$. Whereas d and $\boldsymbol{n}$ denote the Cartesian z-distance and the surface unit normal of the current input point, respectively. As seen previously, the z-distance cannot exceed 10 meters, and a valid range measurement of depth camera usually occurs when the surface normal describes an oblique angle smaller than 80° with the camera direction [5]).

[1] $\lfloor \cdot \rceil$ refers to the closest integer rounding operation.

Thus, a maximum radius size of an input point is defined by $r_{\max} = \frac{5\sqrt{2}}{\cos 80^\circ} \cdot \frac{\max(s_x,s_y)}{f}$. Additionally, we can consider the intrinsic parameter's ratio $\frac{\max(s_x,s_y)}{f}$ to be in any case smaller than $1/200$ which leads to a maximum radius size of $r_{\max} \approx 0.2$ meters, which is a quite conservative upper bound for real-world applications. Thus, we encode the radius as a `uint16` scalar giving $r_e = \lfloor 327.675 \cdot r \rceil$.

In summary, the proposed encoding results in a storage of 2 `float32`, 3 `uint16` and 1 `uint8` scalars (120 bits + **8 bits alignment cut-off**) for the set of point properties in contrast to the naive storage of 9 `float32` scalars (288 bits), which leads to a compression ratio of $1:2.25$. We show that our compression scheme leads to a negligible difference to the original point-based fusion method (see below for a detailed evaluation).

V. Results

In order to evaluate the proposed method, we use four different data sets. Two real-world data sets are used to evaluate the proposed compression method without storing or processing the anisotropy. Two simulated data sets are used to obtain a quantitative comparison of the isotropic reconstruction with our novel anisotropic accumulation scheme (with compression enabled in both instances).

Totempole: This data set is provided by Zhou and Koltun [16] and consists of 8,853 RGB-D frames ($\approx$ 5 minutes of acquisition time) from a Kinect-like camera. Fig. 2 shows the reconstructed scene given by our framework. Note that for this data set only pseudo-groundtruth of camera pose and geometry is given, based

on the approach by Zhou and Koltun. (Available on http://web.stanford.edu/~qianyizh/projects/scenedata.html)

Office: This data set is provided by Kerl et al. [19] and contains 2,509 RGB-D frames (≈ 1.4 minutes of acquisition time) given by a Kinect-like camera, see Fig. 4. The data set includes the camera path groundtruth acquired by an infrared tracking system and was designed for SLAM benchmark applications. (Available on http://vision.in.tum.de/data/datasets/rgbd-dataset)

Buddha: This data set is generated using our Time-of-Flight simulator, which is an enhanced version of Keller and Kolb [20], applied to the Stanford Buddha model scaled to 3 meters height. It is composed of 237 depth frames disturbed with Gaussian noise on the computed polar distance using the formulation of Nguyen et al. [12] for the Kinect structured-light camera. This formulation relates the standard deviation of the z-distance noise to the measured distance via a second-degree polynomial and was modeled using images of planar regions located at different distances.

Statue: This second simulated data set is generated in the same way as the Buddha scene and consists of 286 frames.

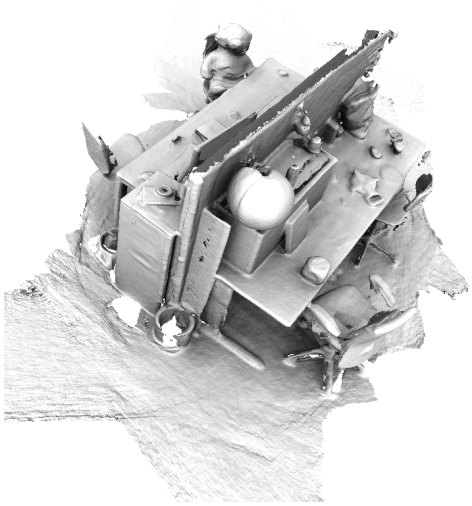

Fig. 4. Overview of the *Office* scene data set from [19].

A. Encoding Evaluation

In order to evaluate our compression representation, we use two data sets given by the *Totempole* and the *Office* scenes. The following three representations are compared to each other:

naive storage: refers to the original point-based fusion framework [5] (uncompressed model).

enc16: compresses normals and viewing rays in our low-resolution, 16-bit representation.

enc32: compresses normals and viewing rays in our high-resolution, 32-bit representation.

The *Totempole* data set is used to highlight the visual quality and the storage ratio. We showed that the proposed compression scheme retains the visual reconstruction quality if the `enc32` compression is used for unit vector representations; see Fig. 3. Concerning the storage gain, the final *Totempole* reconstructed model is composed of 7,822,519 oriented points. The naive storage method (9 `float32` scalars) will lead to a memory usage of 269 MB where our new compression scheme leads to a memory usage of 104.4 MB (**+ 7.5 MB alignment cut-off**). However, our method requires the storage of all the camera poses (8,853 $\times$ 12 `float32` scalars) which enlarges the memory footprint by 415 KB, i.e., by 0.4%.

The *Office* data set is used in order to quantitatively evaluate the compression scheme against the camera tracking and the reconstructed geometry quality. Fig. 5 shows the camera center position errors computed by the Iterative Closest Point (ICP) algorithm for the naive storage, the `enc16`, and the `enc32` encoding schemes. Whereas the `enc16` encoding leads to a higher error in terms of the camera pose estimation, the `enc32` encoding gives camera pose errors very close to the uncompressed method. Additionally, we evaluate the quality of the geometry model reconstructed by each compression scheme. Since no geometry groundtruth is given, we generated a pseudo-groundtruth by applying our reconstruction framework without compression using the groundtruth camera poses. This pseudo-groundtruth is compared to three different reconstruction methods that all use the ICP algorithm to track the camera motion. Fig. 6 shows the Euclidean distance errors of the `enc16`, `enc32` and uncompressed storage. Note how negligible the difference is between the `enc32`, and the naive storage. For a better view on the distance error statistics comparison, refer to Tab. I.

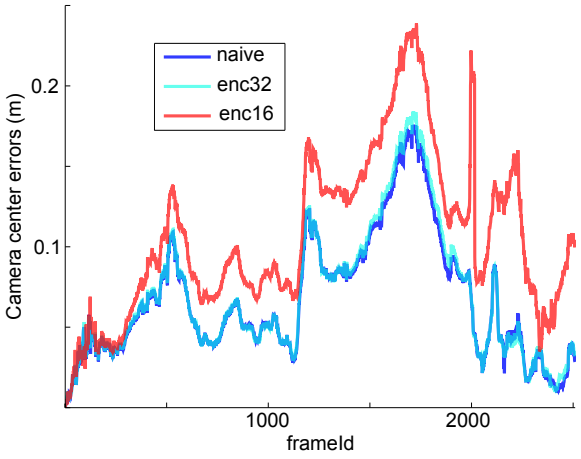

Fig. 5. Camera position errors using the pose groundtruth with the naive representation and the two different compression schemes for the *Office* data set.

B. Anisotropic Fusion Evaluation

In order to evaluate the benefit of the anisotropic fusion, it is important to have proper groundtruth of the scene geometry. Therefore, we used the simulated data sets, i.e., the *Buddha* and the *Statue* scenes. We run our approach for two different scenarios processing the full depth sequences with known groundtruth camera poses. In order to evaluate the anisotropic

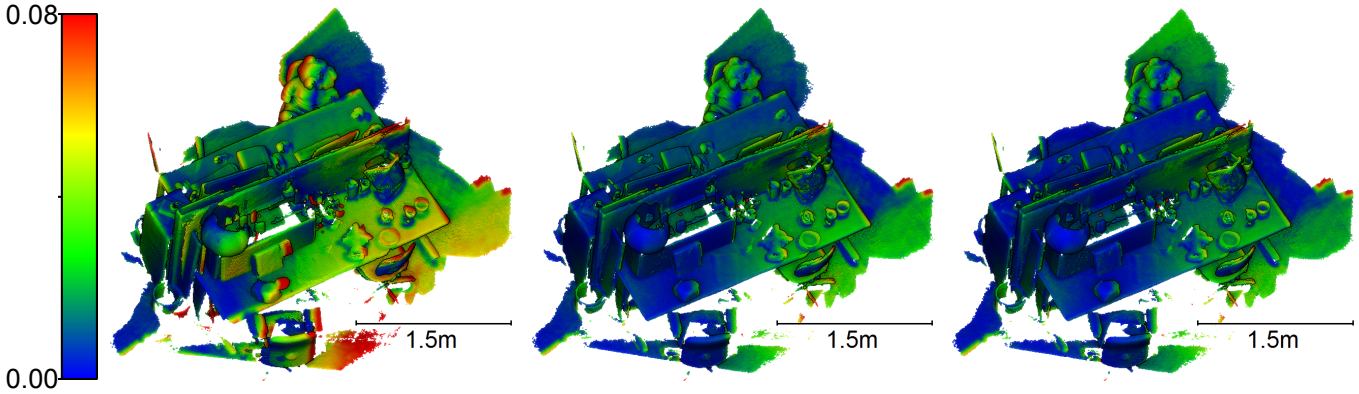

Fig. 6. Color-coding of the geometry distance errors of the *Office* scene for different compression schemes. The reconstruction using the `enc16` (left), the uncompressed (center) and the `enc32` encoding (right). The images are generated using the CloudCompare software [21].

Methods	**enc16**	**naive**	**enc32**
Error Distances mean±std (mm)	24.0±18.3	12.2±10.31	13.0±10.6

TABLE I
DISTANCE ERROR STATISTICS FOR THE *Office* SCENE EXPERIMENT.

fusion, we use the groundtruth camera poses given by our simulator in order to avoid any external error introduced by the ICP algorithm. First, the data is processed using a simple isotropic fusion as it is commonly done for KinectFusion-like approaches. Whereas the other scenario consists of processing the data sequence with anisotropic fusion. Both resulting point clouds are compared to the groundtruth mesh. For each point, the minimal distance error to all mesh faces is computed.

Fig. 9 (left) shows a close view of the point distance errors for the isotropic case, and 9 (right) concerns the anisotropic fusion for the *Buddha* scene. One can clearly see that the anisotropic fusion noticeably reduces the overall point distance errors. Fig. 7 shows the statistic of the errors depending on the confidence counter attribute, i.e., the number of point merges. For the isotropic case, the distance error of model points with a confidence counter greater than 30 is increasing. Fig. 10 visualizes these points and their distance errors, which are mainly located around the lower part of the Buddha. Due to the specific camera path, this region of the scene has been observed by many frames with a comparably large range noise. Apparently, the isotropic accumulation has more difficulties with this strong anisotropy than our anisotropic approach. The total mean distance errors is 1.67 ± 1.4249 mm for the isotropic fusion whereas the anisotropic fusion leads to a total mean of 1.4856 ± 1.3452 mm.

The simulated *Statue* scene confirms our observation, even though the increase of quality is less significant than for the Buddha scene. The error statistics in Fig. 8 show a comparable error statistics for points up to 30 merges and again an improvement beyond 30 merges. The points with a confidence counter greater than 30 are shown in Fig. 11.

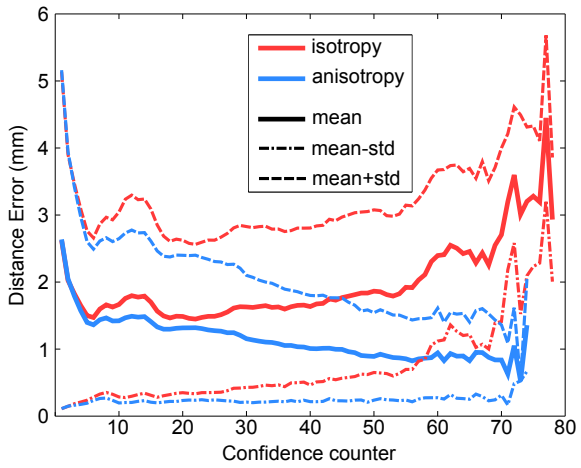

Fig. 7. Comparison of distance error statistics of the *Buddha* scene for the isotropic and anisotropic accumulation. The mean and the standard deviation are plotted. The confidence counter is related to the number of merges for the model points.

C. Performance

We demonstrate the efficiency of our method by evaluating the performance of the different compression schemes and the anisotropy in isolation. Tab. II shows a detailed summary of the timings. Note how the compressed encoding is faster than the original method for the generation of model maps. This is easily explained by the fact that loading the compressed point attributes into a vertex buffer requires 4 floats, whereas the naive storage requires 9 floats per point. Furthermore, the anisotropy is not used during this processing which explains the similar timing with the one from the compression alone.

VI. CONCLUSION

In summary, we proposed a new efficient point-based reconstruction framework that allows a better handling of anisotropic noise of range camera. We introduce a point attributes compression scheme that allows large-scale reconstruction reducing the final storage by half with the same

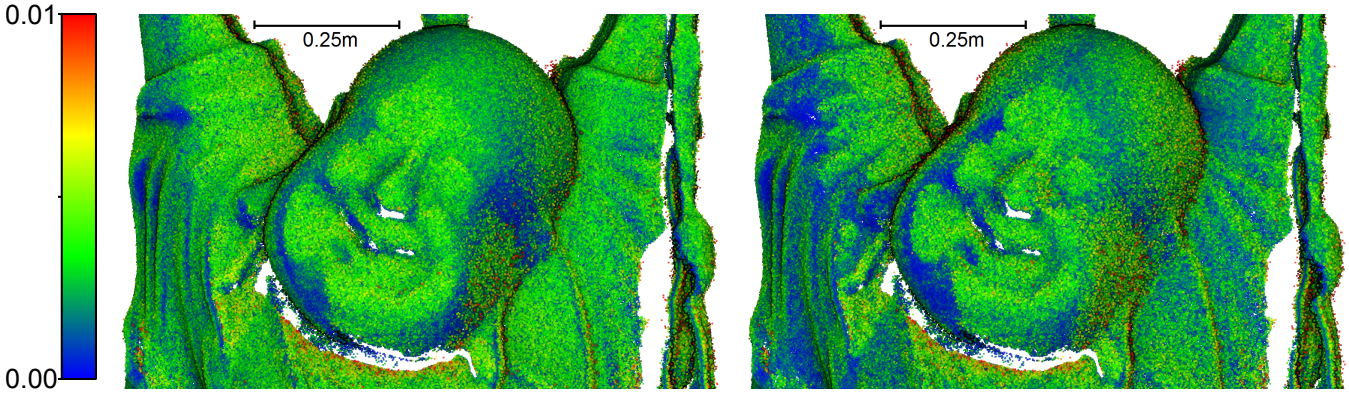

Fig. 9. Color-coded error distances of our *Buddha* scene. The point distance errors to the groundtruth mesh for the isotropic fusion *(left)* and for the anisotropic fusion *(right)*, The images are generated using the CloudCompare software [21].

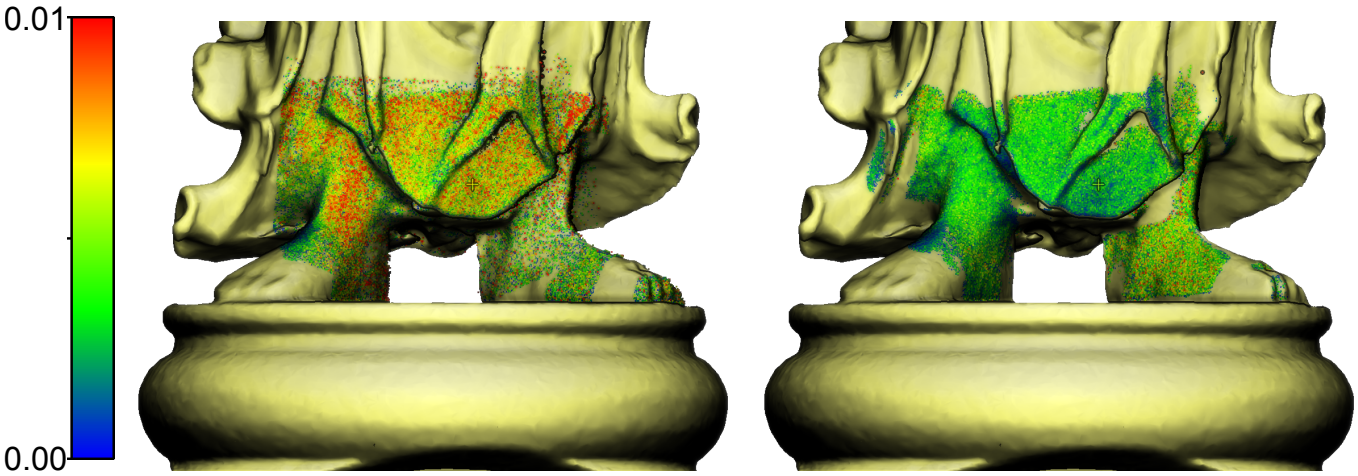

Fig. 10. Color-coded error distances of the *Buddha* scene in a region with high anisotropy. Here only points that have a confidence counter greater than 30 are shown. The anisotropic accumulation *(right)* better handles this region with strong distance noise compare to the isotropic fusion *(left)*. The images are generated using the CloudCompare software [21].

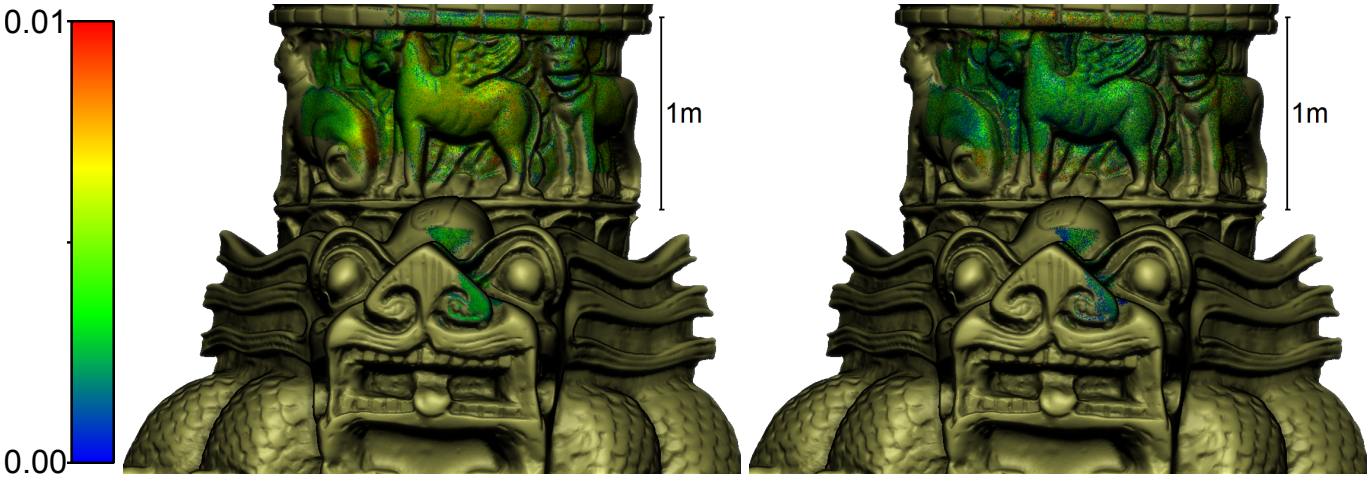

Fig. 11. Color-coded error distances of the *Statue* scene in a region with high anisotropy. Here only points that have a confidence counter greater than 30 are shown. The anisotropic accumulation *(right)* better handles this region with strong distance noise compare to the isotropic fusion *(left)*. The images are generated using the CloudCompare software [21].

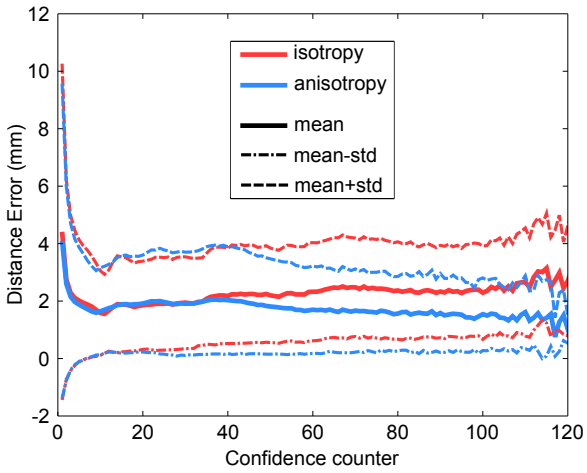

Fig. 8. Comparison of distance error statistics of the *Statue* scene for the isotropic and anisotropic accumulation. The mean and the standard deviation are plotted. The confidence counter is related to the number of merges for the model points.

Methods (all times in msec)	**MapComp** min, max mean±std	**IdxMap** min, max mean±std	**Accum** min, max mean±std	**GenModelMaps** min, max mean±std
Naive	1.9, 6.1 3.1±0.5	0.7, 2.6 1.6±0.5	3.2, 6.7 5.1±0.3	2.1, 6.7 5.0±1.2
Encoding	1.9, 6.1 3.1±0.5	0.6, 2.6 1.4±0.4	3.4, 7.9 5.2±0.4	1.6, 4.4 3.4±0.5
Encoding + Anisotropy	1.9, 6.7 3.1±0.6	0.6, 2.6 1.8±0.4	3.5, 8.0 5.6±0.4	1.6, 4.4 3.4±0.5

TABLE II

TIMINGS GIVEN BY THREE METHODS USING THE *Buddha* SCENE FOR FOUR PROCESSING MODULES. **MAPCOMP** (DEPTH MAP PREPROCESSING), **IDXMAP** (INDEX MAP GENERATION), **ACCUM** (DEPTH MAP FUSION), **GENMODELMAPS** (RENDERING). RED COLORS REFERS TO THE MODULES WHERE THE ANISOTROPIC INFORMATION IS USED.

performance. We demonstrate that this encoding does not disturb neither the camera tracking algorithm nor the quality of the reconstructed geometry. Furthermore, we demonstrate that anisotropic fusion improves the overall quality of the reconstruction.

ACKNOWLEDGMENT

This work was funded by the German Research Foundation (DFG) as part of the research training group GRK 1564 *Imaging New Modalities*, and by the UK Engineering and Physical Sciences Research Council (grant EP/K023578/1).

REFERENCES

[1] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D model acquisition," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 21, no. 3, pp. 438–446, 2002.

[2] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. IEEE Int. Symp. Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[3] H. Roth and M. Vona, "Moving volume KinectFusion," in *British Machine Vision Conf.*, 2012, pp. 1–11.

[4] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 169, 2013.

[5] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D reconstruction in dynamic scenes using point-based fusion," in *Proc. Int. Conf. 3D Vision (3DV)*, 2013, pp. 1–8.

[6] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Robotics-DL tentative*. International Society for Optics and Photonics, 1992, pp. 586–606.

[7] D. Falie and V. Buzuloiu, "Noise characteristics of 3d time-of-flight cameras," in *Proc. Int. Symp. Signals, Circuits and Systems (ISSCS)*, vol. 1, 2007, pp. 1–4.

[8] F. Lenzen, K. I. Kim, H. Schäfer, R. Nair, S. Meister, F. Becker, C. S. Garbe, and C. Theobalt, "Denoising strategies for time-of-flight data," in *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer, 2013, pp. 25–45.

[9] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real time motion capture using a single time-of-flight camera," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 755–762.

[10] L. Maier-Hein, A. M. Franz, T. R. dos Santos, M. Schmidt, M. Fangerau, H. Meinzer, and J. M. Fitzpatrick, "Convergent iterative closest-point algorithm to accomodate anisotropic and inhomogenous localization error," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 8, pp. 1520–1532, 2012.

[11] M. Mittring, "Finding next gen: Cryengine 2," in *ACM SIGGRAPH 2007 courses*, 2007, pp. 97–121.

[12] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3D reconstruction and tracking," in *Proc. Int. Conf. 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, 2012, pp. 524–530.

[13] C. Kerl, M. Souiai, J. Sturm, and D. Cremers, "Towards illumination-invariant 3d reconstruction using ToF RGB-D cameras," in *Proc. Int. Conf. 3D Vision (3DV)*, 2014, pp. 39–46.

[14] Z. H. Cigolle, S. Donow, D. Evangelakos, M. Mara, M. McGuire, and Q. Meyer, "A survey of efficient representations for independent unit vectors," *Journal of Computer Graphics Techniques*, vol. 3, no. 2, 2014.

[15] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, "Dense planar SLAM," in *Proc. IEEE Int. Symp. Mixed and Augmented Reality (ISMAR)*, 2014, pp. 157–164.

[16] Q.-Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 112, 2013.

[17] E. Praun and H. Hoppe, "Spherical parametrization and remeshing," in *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, 2003, pp. 340–349.

[18] T. Weise, T. Wismer, B. Leibe, and L. Van Gool, "In-hand scanning with online loop closure," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, 2009, pp. 1630–1637.

[19] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 2100–2106.

[20] M. Keller and A. Kolb, "Real-time simulation of time-of-flight sensors," *J. Simulation Practice and Theory*, vol. 17, pp. 967–978, 2009.

[21] D. Girardeau-Montaut, "CloudCompare OpenSource Project," 2013. [Online]. Available: http://www.danielgm.net/cc/